

## 8) Polymorphism: Run time



Take decision at run time.

Vehicle \*v2 = new car;

v2 → print() → vehicle()

---

v2 → print() ~~also~~ → To get car class print() function we need

\* To achieve Run time polymorphism we can use Virtual Functions.

```
class Vehicle {
```

```
public:
```

```
    string color;
```

```
    virtual void print() {
```

```
        cout << "Vehicle" << endl;
```

```
}
```

Ex:

```
class car : public Vehicle {
```

```
public:
```

```
    int num Gears;
```

```
    void print() {
```

```
        cout << "Car" << endl;
```

```
}
```

∴

Now when we print

~~Vehicle~~  $V_2 = \text{new car}$

$V_2 \rightarrow \text{print}() \rightarrow$  now it ~~finds~~ call car class print

\* If  $\text{print}()$  function is not defined in car class, then automatically the base class ~~car~~ print function is called.

\* If  $\text{print}$  function is not defined in vehicle, then it raises an error.

\* Run time polymorphism is achieved using virtual functions

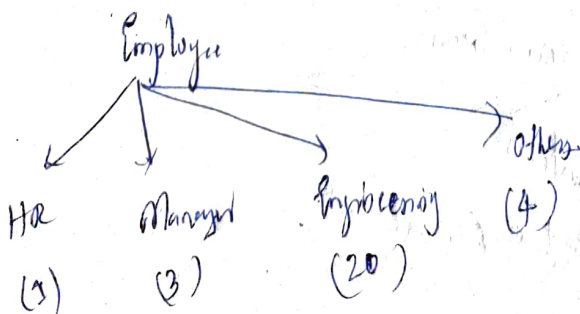
Use case of Run time polymorphism

1) Organisation:

Calculating salaries:

→ HR  
- Manager  
- Engineers  
- Others

} These have different salaries



Every child class has a function called calculateSalary().

Employee \* \* c      ~~new Employee \* [20]~~  
new Employee \* [20]